AD-A049 684 STANFORD UNIV CALIF DEPT OF OPERATIONS RESEARCH F/G 12/1
TWO-PHASE ALGORITHM FOR NONLINEAR CONSTRAINT PROBLEMS.(U)
AUG 77 J B ROSEN N00014-75-C-0267
TR-77-22 NL

UNCLASSIFIED

| OF |
AD
A049684

END
DATE
FILMED
3-78
DDC

TWO-PHASE ALGORITHM FOR NONLINEAR CONSTRAINT PROBLEMS

by

J.B. Rosen

TECHNICAL REPORT 77-22

August 1977

DEPARTMENT OF OPERATIONS RESEARCH

Stanford University
Stanford, California

Research and reproduction of this report were supported by the Office of Naval Research Contract N00014-75-C-0267 in the Department of Operations Research. This report was also issued as Technical Report No. 77-8 in the Computer Science Department of the University of Minnesota, prepared under National Science Foundation Grant MCS 76-23311.

Two-Phase Algorithm for Nonlinear Constraint Problems

J. B. Rosen
Computer Science Department
University of Minnesota

Abstract

An algorithm is described which solves the general nonlinear programming problem with nonlinear constraints. The computational implementation is based on the iterative use of any available package which solves the linearly constrained problem with a nonlinear objective function. Large, sparse problems with nonlinear constraints can be solved by this algorithm, provided a suitable linear constraint package is used.

The algorithm consists of two phases. The first (Phase I) uses an external squared penalty function to find a point x^1, close to a local minimum. Starting with x^1, the algorithm then solves a sequence of linearly constrained problems (Phase II). Selected nonlinear constraints are linearized for each such Phase II iteration. With suitable assumptions, convergence from any initial point, with quadratic convergence in Phase II, is shown.

The practical implementation of this algorithm is described, and its potential application to a model for the assessment of energy alternatives is discussed briefly.

1. Introduction

Nonlinear programming is now being used effectively to solve relatively large computer models for the assessment of various energy alternatives. In a recent study of energy-economy interactions over a 75 year period [4], a model was developed which consisted of approximately 350 rows, 600 columns, and an objective function in which 80 variables appear in a nonlinear manner. The coefficient matrix was somewhat less than 1% dense. A variety of cases were run, where each such run required 30 to 90 seconds of computer time on an IBM 370/168. The nonlinear aspects of this model appear to be crucial to a realistic representation of the economy-wide production function and the consumption-investment tradeoffs. It also seems that an attempt to represent this system in terms of an entirely linear model would have led to a larger and more cumbersome problem.

The solutions were obtained using the package MINOS [6,7], which can be used for large, sparse problems with a nonlinear objective function, but linear constraints only. The original model formulation contained 25 nonlinear constraints. Fortunately, for the purposes of the study, it was known that they were equality constraints, and each such nonlinear equation was used to eliminate a variable from the original objective function. However, this technique will not generally be valid. Therefore an efficient computer program capable of solving large, sparse problems with nonlinear inequality constraints is definitely needed for this class of problems, as well as many others.

The work described in this report was largely motivated by this need. An important objective was to develop a method for solving problems with nonlinear constraints, taking full advantage of available computer packages for solving the linearly constrained problem. This is accomplished by the Two-Phase algorithm*, which uses such a package iteratively to solve a sequence of linearly constrained problems. During each iteration the package is treated as a "black box", with only the usual information (linear constraint matrix coefficients, objective function and gradient subroutines, and an initial point) being supplied to it. Thus the most suitable existing package can be used, and if an improved package for linearly constrained problems becomes available, it can replace the existing package with a minimum of effort.

---

* A preliminary version of the Two-Phase algorithm was presented earlier [14].

The general problem considered is the following. Let $S \subset R^n$ denote a convex polyhedron defined by the linear constraints and bounds:

$$S = \left\{ x \;\middle|\; \begin{array}{l} Ax = b \\ \underline{b} \leq x \leq \bar{b} \end{array} \right\}$$

where the vectors b, $\underline{b}$, $\bar{b}$ and the elements of the matrix A are specified. To simplify the presentation, it is assumed that all linear inequalities have been converted to equations in the standard manner. Also, some (or all) variables may be unbounded ($\underline{b}_i = -\infty$ or $\bar{b}_i = \infty$, or both). A set of q+1 functions $\phi_i(x)$, $i = 0,1,\ldots,q$, are defined on S and (at least for the validity of convergence proofs) are assumed to have continuous second derivatives for $x \in S$. The function $\phi_0(x)$ is the objective function, and the ϕ_i, $i = 1,\ldots,q$, are the nonlinear constraint functions. The domain $V \subset S$ is now given by

$$V = \left\{ x \;\middle|\; \begin{array}{l} x \in S \\ \phi_i(x) \leq 0, \; i = 1,\ldots,q \end{array} \right\}$$

It is assumed that V is nonempty. Again to simplify matters, it is assumed that all nonlinear constraints are inequalities. Nonlinear equality constraints are handled as a special case with no difficulty, as discussed later. The general problem to be solved is then

$$\text{(PN)} \quad \min_{x \in V} \phi_0(x)$$

The theoretical basis for the solution of (PN) by the Two-Phase algorithm is given in the next two sections. It is shown there that with suitable assumptions, the algorithm will give global convergence to at least a local minimum x* of (PN). From an arbitrary initial point x^0, a point x^1 close to x*, is obtained in Phase I by minimizing an external squared penalty function subject to $x \in S$. An estimate λ^1 is also obtained of the optimal vector of multipliers λ^*, corresponding to x*.

Starting with x^1 and λ^1, the Phase II iterations give a sequence of vectors $\{x^k\}$ and $\{\lambda^k\}$. In the k^{th} iteration the nonlinear constraints are

linearized about x^k and the minimum is found to a linearly constrained problem with a modified Lagrangian as the objective function. The solution to this problem gives x^{k+1} and λ^{k+1}. The vectors thus obtained satisfy the relation $x^{k+1} \in M(x^k)$, where M is a point-to-set mapping. It is shown that a fixed point of M is a Kuhn-Tucker point of (PN). Furthermore, with some additional assumptions it is shown that the sequence $\{x^k, \lambda^k\}$ converges to (x^*, λ^*) at a quadratic rate. This theoretical justification for the Two-Phase algorithm draws heavily on earlier work on external penalty functions [2], and the iterative linearization of the nonlinear constraints [9,12]. For a recent discussion of methods using constraint linearization, and comparison with augmented Lagrangian methods, see [8].

The Two-Phase algorithm can be implemented using any computer package which solves the linearly constrained problem with a nonlinear objective function. Several suitable packages for linearly constrained problems are now available [1,3,7,13]. The Two-Phase algorithm has already been implemented using the GPM package, and is being implemented using MINOS.

The capabilities of the resulting computer program for solving (PN) will be determined largely by those of the package used. Thus the size and structure of problems which can be solved, and the efficiency with which they are solved, will depend primarily on the package used in the implementation. The size and structure of the problem is essentially determined by that of the original linearly constrained domain S, since in general, the additional linearized constraints added in Phase II will not increase the problem size significantly. For example, if the matrix A which defines S is large and sparse, then the linearly constrained problems solved in both Phase I and Phase II will also be large and sparse, and will therefore be solved most efficiently by a package designed for such problems.

The initial implementation of the Two-Phase algorithm has been completed using the GPM package which is suitable for relatively small, dense problems consisting of no more than 40 variables and 80 linear inequality constraints This implementation is called GPM/NLC. It has the advantage that it is convenient to use for relatively small problems, and it therefore served as an excellent vehicle for developing and testing the practical implementation of the Two-Phase algorithm. Using this package it was convenient to experiment with a number of possible modifications in order to improve the efficiency of the resulting computer program.

A variety of relatively small test problems have been run using GPM/NLC.

These problems all consisted of 15 or fewer variables, and a maximum of 20 constraints, plus bounds. All problems contained nonlinear inequality constraints, both convex and nonconvex. Some also had nonlinear equality and linear constraints. A number of these problems have been used previously as test problems for other methods [16]. In all cases the performance by GPM/NLC was comparable, and often better, than that of other methods on the same problems. This performance depends on the proper choice of the penalty parameter μ in Phase I, and on the quadratic convergence rate achieved in Phase II. Normally convergence is attained with no more than five iterations in Phase II, with the final two of these relatively fast, since the corresponding change in x is small.

A preliminary implementation of the Two-Phase algorithm using MINOS has also been tested. This implementation is called MINOS/NLC. These tests have used a modified version of the energy alternative model discussed above with 350 linear constraints. In addition a total of 30 nonlinear inequality constraints have been added, only some of which are active at the optimum. Based on these preliminary tests it is estimated that nonlinear constraint problems of this size can be solved in 3 to 5 minutes on an IBM 370/168, or comparable machine. Details of the computational results obtained with GPM/NLC and MINOS/NLC will be given elsewhere [15].

2. Linearization of Nonlinear Constraints

We first consider the linearly constrained minimization problem

$$(PS) \quad \min_{x \in S} \phi_0(x)$$

Since S is bounded, there is a point $\bar{x}^* \in S$ at which $\phi_0(x)$ attains its global minimum; that is, $\phi_0(\bar{x}^*) \leq \phi_0(x)$ for all $x \in S$. It follows directly from the previous assumptions that a global minimum of (PN) is attained at some point $x^* \in V$, which may not, however, be unique. Since $V \subset S$, we have $\phi_0(\bar{x}^*) \leq \phi_0(x^*)$. A special situation occurs if $\bar{x}^* \in V$, in which case we have $x^* = \bar{x}^*$ (all the nonlinear constraints are redundant).

While we would like to determine a global minimum of (PN), we will frequently only find a local minimum. Provided that the ϕ_i satisfy some regularity condition, a local minimum will always be a Kuhn-Tucker point (i.e., the first-order Kuhn-Tucker optimality conditions will be satisfied).

Now given any point $y \in S$, we consider the linearized approximations to the constraints $\phi_i(x) \leq 0$. That is, we replace them with

$$h_i(y;x) \equiv \phi_i(y) + (x-y)^T \nabla \phi_i(y) \leq 0, \quad i = 1,\ldots,q$$

This gives us a polyhedral set $W(y) \subset S$ defined by

$$W(y) \equiv \left\{ x \;\middle|\; \begin{array}{l} x \in S \\ h_i(y;x) \leq 0, \quad i = 1,\ldots,q \end{array} \right\}$$

Note that if, say $\phi_j(x) = 0$, we use the corresponding linear equality $h_j(y;x) = 0$ in $W(y)$. For an arbitrary $y \in S$, $W(y)$ could be empty. However, $W(y)$ is not empty for any $y \in V$, since $y \in V$ implies $y \in W(y)$. It is convenient to consider W as a point-to-set mapping $W: S \to S$. We will make an additional assumption later about the functions ϕ_i, to insure the continuity of the mapping W.

The linearized constraints give us a problem which is closely related to (PN). Given any $y \in S$ we consider the problem

$$(PL) \quad \min_{x \in W(y)} \Psi(y;x)$$

where $\Psi(y;x) \equiv \phi_0(x) + \sigma(y;x)$, and $\sigma(y;x)$ satisfies the relations $\sigma(y;y) = 0$ and $\nabla\sigma(y;y) = 0$. Note that this problem consists of linear constraints and bounds only, with the nonlinearity confined to the objective function.

As with (PN) we would like to find a global minimum of (PL), but we will often find only a local minimum. We therefore interpret an optimal solution to (PL) as any point $x \in W(y)$ at which $\Psi(y;x)$ has a local (constrained or unconstrained) minimum. In general, the word minimum will mean a local minimum, unless global minimum is explicitly stated.

The problem (PL) will always have an optimal solution, provided that $W(y)$ is not empty. The problem (PL) therefore defines another point-to-set mapping $M:S \to S$. Specifically, $M(y) \subset W(y)$ is the point (or set of points) at which $\Psi(y;x)$ attains its minimum for $x \in W(y)$. That is

$$M(y) \equiv \arg\min_{x \in W(y)} \Psi(y;x)$$

Our main interest is in a fixed point of this mapping, that is a point y such that $y \in M(y)$.

Theorem 1

A fixed point of the mapping M is a Kuhn-Tucker point of (PN).

Proof: Let y be a fixed point of M. Then we have $y \in W(y)$. We first show that $y \in V$. If $y \notin V$, then for at least one i, say $i = k$, we must have $\phi_k(y) > 0$. But then we would have $h_k(y;y) = \phi_k(y) > 0$, so that $y \notin W(y)$.

Since y is an optimal point for the problem (PL) with all constraints linear, it follows that the first-order Kuhn-Tucker optimality conditions are satisfied at y. These conditions involve the functions $h_i(y;x)$ and their gradients $\nabla h_i(y;x)$ evaluated at $x=y$, for all active linearized constraints at y, that is, those for which $h_i(y,y)=0$. The active linear constraints and bounds, as well as the objective function gradient $\nabla\Psi(y;y)$, also enter into these conditions. But these are just the Kuhn-Tucker optimality conditions for the problem (PN) at the point $x=y$, since we have $h_i(y;y) = \phi_i(y)$, $\nabla h_i(y;y) = \nabla\phi_i(y)$, and $\nabla\Psi(y;y) = \nabla\phi_0(y)$. ■

It should be noted that we also have the same value for the objective functions of (PL) and (PN) at a fixed point y, since $\Psi(y;y) = \phi_0(y)$.

We now consider using the mapping M as the essential part of an algorithm to solve (PN). For this purpose we desire an algorithm which generates a sequence of points converging to a fixed point of M. We consider the following.

Algorithm A

Starting with an initial point $x^0 \in S$, obtain the sequence of points $\{x^k\}$, generated by (PL). For this sequence we have

$$(1) \quad x^{k+1} \in M(x^k) , \quad k = 0,1,\ldots$$

The convergence of this algorithm to a fixed point will, in general, depend on the use of an appropriate term $\sigma(y;x)$ in the objective function. It is, however, of some interest to show that for a suitably restricted class of problems this algorithm converges to a fixed point of M, starting with any $x^0 \in V$, even for $\sigma \equiv 0$.

Assume that the ϕ_i are concave on S, and that they satisfy a regularity condition which ensures that the mapping W is continuous [5,10]. Note that for concave ϕ_i the feasible set V will not, in general, be convex, and in fact can be called reverse-convex [5]. For an earlier discussion of this case see [11].

Lemma 1 Starting with any $x^0 \in V$, generate a sequence $\{x^k\}$ by means of Algorithm A, with $\sigma \equiv 0$. Then for $k = 0,1,\ldots$, we have $x^k \in V$ and $\phi_0(x^{k+1}) \leq \phi_0(x^k)$. Furthermore, there will be a convergent subsequence which converges to a fixed point of M.

Proof: For ϕ_i concave we have $\phi_i(x) \leq h_i(y,x)$ for any $x,y \in S$. Then for any $x \in W(y)$ we have $h_i(y;x) \leq 0$, $i = 1,\ldots,q$, which implies $\phi_i(x) \leq 0$, $i = 1,\ldots,q$. Thus $y \in W(y) \subset V$, for any $y \in V$.

Since $V \subset S$ is compact, the sequence $\{x^k\}$ has an accumulation point, say $x^* \in V$. Then because of monotonicity and continuity, $\phi_0(x^*) \leq \phi_0(x^k)$ for all k. Now suppose x^* is not a fixed point of M. Then there must exist a point $\bar{x} \in W(x^*)$, with $\bar{x}$ close to x^*, such that $\phi_0(\bar{x}) < \phi_0(x^*)$. By the continuity of the mapping M there then would be a point x^k of the sequence, sufficiently close to x^* such that $\phi_0(x^{k+1}) < \phi_0(x^*)$, a contradiction. ■

The well-known difficulty with Algorithm A for $\sigma = 0$, is that it will not, in general, solve (PN) for the standard convex problem (all ϕ_i convex),

where any Kuhn-Tucker point is a global minimum. This can be easily seen by applying it to the problem in R^2:

$$\min_{x_1,x_2} \left\{ -x_1 - x_2 \;\middle|\; \begin{array}{l} 0 \le x_1,\ x_2 \le 2 \\ x_1^2 + x_2^2 \le 2 \end{array} \right\}$$

with the optimum solution $x_1^* = x_2^* = 1$. For any starting point not on the line $x_1 = x_2$, the sequence of points generated will all lie on the boundaries of the square.

This difficulty can be overcome by a suitable choice for σ. Replacing ϕ_0 by the Lagrangian [12], or a modified Lagrangian [9] will ensure convergence (in fact, at a quadratic rate) provided the starting point is close enough to a local minimum. The modified Lagrangian will be used here. This is obtained by choosing

$$(2) \quad \sigma(y;x) = \sum_{i=1}^{q} \lambda_i(y)[\phi_i(x) - h_i(y;x)]$$

where the Lagrange multipliers λ_i are determined in an appropriate manner. More specifically, the solution of (PL) with $y=x^k$, and $\lambda_i(x^k) = \lambda_i^k$, gives a point $x^{k+1} \in M(x^k)$ and also a set of multipliers $\lambda_i^{k+1} \ge 0$, $i = 1,\ldots,q$, corresponding to the linearized constraints. Only those multipliers which correspond to an active linearized constraint can be positive, and therefore contribute to σ.

We consider a local minimum x^* of (PN) and let $\lambda^* \ge 0$ be the corresponding vector of multipliers. We make a regularity assumption about the constraints ϕ_i at x^* so that x^* is a Kuhn-Tucker point, and also assume that second-order sufficiency conditions hold there. Some additional assumptions are needed to insure that certain matrices (involving the Lagrangian and its derivatives) are bounded in the neighborhood of (x^*,λ^*).

Lemma 2. Apply Algorithm A, with σ as given by (2) and $y=x^k$ at the k^{th} iteration, to generate a sequence of points $\{x^k\}$ and corresponding vector of multipliers $\{\lambda^k\}$, $k = 2,3,\ldots$, starting with specified vectors x^1 and λ^1. Then if

$$(3) \quad ||(x^1,\lambda^1) - (x^*,\lambda^*)|| \le \rho \le 1/2\alpha$$

the sequence $\{x^k, \lambda^k\} \in R^{n+q}$ will converge to (x^*, λ^*) at a quadratic rate. The constants ρ and α depend only on the norm of certain matrices evaluated in the neighborhood of (x^*, λ^*).

More specifically, we have

$$(4) \quad ||f(z^{k+1})|| \leq 2(\tfrac{1}{2})^{2^k} ||f(z^1)||$$

$$(5) \quad ||\Delta z^{k+1}|| \leq \alpha ||\Delta z^k||^2$$

$$(6) \quad ||z^k - z^*|| \leq \frac{2}{\alpha}(\tfrac{1}{2})^{2^k}$$

for $k = 1,2,\ldots$. In these relations we simplify notation by using $z \equiv (x,\lambda) \in R^{n+q}$, $\Delta z^k \equiv z^{k+1} - z^k$. The vector function $f(z)$ is given by

$$f(z) \equiv \begin{pmatrix} \Lambda'(x,\lambda) \\ w(x,\lambda) \end{pmatrix} \in R^{n+q}$$

where $\Lambda(x,\lambda) = \phi_0(x) + \Sigma\lambda_i\phi_i(x)$ is the Lagrangian for (PN), Λ' is its gradient with respect to x, and $w^T(x,\lambda) \equiv (\lambda_1\phi_1(x),\ldots,\lambda_q\phi_q(x))$.

The detailed assumptions and proof have been given by Robinson [9], and will not be repeated here. The results hold with $||\cdot||$ representing any consistent norm. To simplify the presentation here, it has been assumed that only nonlinear constraints are active at x^*. If any of the original linear constraints or bounds are active, the gradient $\nabla\phi_0$ must be replaced by its projection in the appropriate subspace.

It should be noted that provided $\lambda_i > 0$ whenever $\phi_i(x) > 0$, the point (x^*,λ^*) is a Kuhn-Tucker point if, and only if, $f(x^*,\lambda^*) = 0$.

Thus from a good starting point Algorithm A gives rapid convergence to the closest local minimum. However, this still does not give us a satisfactory method for getting a Kuhn-Tucker point from a more or less arbitrary given point x^0. To achieve this, we must first generate a good starting point x^1 and a corresponding multiplier λ^1. For this purpose we need a method which generates a point close to a local minimum of (PN), from an initial point x^0 which may lie anywhere in S (the original linearly constrained domain). Since the linearization of the ϕ_i about such an x^0 may give a very poor approximation to the nonlinear constraints, we do not want to use (PL) initially.

3. External Squared Penalty

The most satisfactory answer appears to be the use of an external squared penalty for the nonlinear constraints. That is, we solve the linearly constrained problem

$$(PI) \qquad \min_{x \in S} \bar{\Psi}(\mu;x)$$

where $\bar{\Psi}(\mu;x) = \phi_0(x) + \frac{\mu}{2} \sum_{i=1}^{q} [\phi_i^+(x)]^2$. The $\phi_i^+ = \phi_i$ for $\phi_i > 0$, and are zero otherwise. Nonlinear equality constraints are included by using ϕ_j itself (rather than ϕ_j^+) if $\phi_j(x) = 0$.

The penalty μ must be chosen large enough to insure that x^1 is sufficiently close to x^*, but no larger than necessary, since a large value of μ can greatly increase the computation time of (PI). The solution of (PI) also gives an estimate λ^1 of the optimal vector of multipliers λ^*.

The use of the external squared penalty to solve (PN) has been thoroughly investigated [2]. By choosing a sequence of values $\{\mu^k\} \to \infty$, it is shown that the corresponding sequence of points $\{x^k\}$, given by (PI) converges to a local minimum of (PN). The conditions on (PN) for convergence from an arbitrary initial point appear to be the least restrictive of any globally convergent algorithm. For example, a condition which will ensure global convergence to at least a local minimum of (PN) is that the function $\sum_{i=1}^{q} [\phi_i^+(x)]^2$ be quasiconvex on S. We assume some such condition to hold.

However, instead of solving a sequence of problems (PI) with increasing values of μ, we pick a single (relatively small) value of μ and solve the corresponding problem (PI) to get a point $x^1 = \bar{x}(\mu)$. We call this Phase I.

In general, x^1 will violate some of the nonlinear constraints. We estimate λ^* by computing the vector of multipliers $\lambda^1 = \bar{\lambda}(\mu)$, given by

$$(7) \quad \bar{\lambda}_i(\mu) = \mu \phi_i^+(\bar{x}(\mu)) \geq 0, \; i = 1,\ldots,q.$$

Once we have an x^1 and λ^1 obtained in this way we can use the previous algorithm to obtain rapid convergence to a local minimum. We call this Phase II. The combination of Phase I and Phase II leads directly to:

Algorithm B

1. Start with $x^0 \in S$.

2. Phase I: Solve (PI) with a specified μ, to get x^1 and λ^1 as given by (7). Set $x^k \leftarrow x^1$, $\lambda^k \leftarrow \lambda^1$ and $k \leftarrow 1$.

3. Phase II: Given x^k and λ^k, solve (PL) with σ as given by (2) and $y = x^k$. This gives x^{k+1} and $\lambda^{k+1} \geq 0$.

4. If $||(x^{k+1},\lambda^{k+1}) - (x^k,\lambda^k)|| \leq \varepsilon$, output x^{k+1} and λ^{k+1}. Stop.

5. Set $x^k \leftarrow x^{k+1}$, $\lambda^k \leftarrow \lambda^{k+1}$, $k \leftarrow k+1$, and return to 3.

We will now prove convergence of this algorithm to a local minimum of x^* of (PN) from any point $x^0 \in S$. This proof is based on the assumption that starting from any $x^0 \in S$ the external squared penalty method (PI) converges to a local minimum x^*, as $\mu \to \infty$. The corresponding optimal vector of multipliers will be denoted by λ^*. We also require that the assumptions made prior to Lemma 2 are valid.

Theorem 2

There is a $\bar{\mu} > 0$, such that if we use any $\mu \geq \bar{\mu}$ in Phase I, and starting with $x^0 \in S$ generate a sequence of vectors $\{x^k,\lambda^k\}$, $k = 1,2,\ldots$, by Algorithm B, this sequence will converge to $\{x^*,\lambda^*\}$ with a quadratic convergence rate. For any $\varepsilon \leq \frac{1}{2\alpha}$ Phase II will terminate with vectors x^{k+1} and λ^{k+1} which satisfy

$$(8) \quad ||(x^{k+1},\lambda^{k+1}) - (x^*,\lambda^*)|| \leq \varepsilon$$

Proof: For every $\mu > 0$, (PI) determines an optimal vector $\bar{x}(\mu)$ and the corresponding $\bar{\lambda}(\mu)$ given by (7). By assumption $\lim_{\mu\to\infty} \bar{x}(\mu) = x^*$. As shown in [2], we also have that $\lim \bar{\lambda}(\mu) = \lambda^*$. Thus we can choose $\bar{\mu}$ so that with $x^1 = \bar{x}(\bar{\mu})$ and $\lambda^1 = \bar{\lambda}(\bar{\mu})$, we satisfy the condition (3) of Lemma 2. By Lemma 2, the sequence $\{x^k,\lambda^k\}$ converges quadratically to (x^*,λ^*). At termination we have $||\Delta z^k|| \leq \varepsilon \leq \frac{1}{2\alpha}$. Then using (5), it follows that

$$||(x^{k+1},\lambda^{k+1}) - (x^*,\lambda^*)|| \leq \sum_{j=k+1}^{\infty} ||\Delta z^j|| \leq \varepsilon \sum_{j=0}^{\infty} (\tfrac{1}{2})^{2^j} \leq \varepsilon \quad \blacksquare$$

4. Practical Implementation

In the preceding sections the theoretical basis for the Two-Phase algorithm has been presented. We now consider the practical implementation of this algorithm to give an efficient computer program for finding at least a local minimum of the problem (PN). In addition to efficiency, it is desired to accomplish two related objectives with this implementation.

First, the implementation should be able to use any existing package capable of solving the linearly constrained nonlinear programming problem. This allows the user to take full advantage of the structure of the linear constraint matrix A (relatively small and dense, large and sparse, etc.) by using a suitable package. It also permits the relatively easy substitution of improved packages as they become available. Some existing packages (MINOS in particular) require very little from the user in the way of tolerance or parameter selection. This benefit will then carry over directly to the nonlinear constraint problem, when such a package is used.

A second important objective is a robust program, in the sense that it will find at least a local minimum for a variety of problems typical of those found in practical applications. Typically the objective function and at least some of the constraints will not be convex, only some of the constraints will be active at the minimum, and the initial point may not be feasible. Another practical difficulty will arise if the set S is feasible, but there is no feasible solution to (PN), that is, V is empty. This could happen as a result of conditions which are too stringent, unrealistic assumptions, or an error in formulation. In any event, a practical implementation cannot assume that a feasible solution to V exists, but must have provision to recognize this difficulty, if it occurs, with a minimum of wasted computer time.

The modifications to Algorithm B which are described in this section are intended to achieve these practical objectives. In describing the modified algorithm (to be called Algorithm C) we will assume that the package used to solve the linearly constrained problems has certain desired properties. These are:

1. The ability to find a feasible point with respect to the specified linear constraints and bounds, if such a point exists. If no such point

exists, a "no feasible solution" exit from the package is activated.

2. The package will normally find at least a local minimum for any differentiable objective function, subject to linear constraints and bounds. The optimality test requires that the first order Kuhn-Tucker conditions be satisfied within a certain tolerance. This tolerance will depend on certain computed quantities and may also depend on a user specified tolerance. When the optimality test is satisfied the optimal vector x, and corresponding vector λ of multipliers, is available. If the optimality test is not satisfied after a specified maximum number of iterations, the calculation is stopped, and the current vectors x and λ are available.

3. The package must be able to accept an initial vector x^0, and if x^0 is a local minimum it should recognize this fact without unnecessary computation. If x^0 is not a local minimum it should find a "close" local minimum; that is, if x^0 is feasible it should find a sequence of points with decreasing function values, and if x^0 is not feasible it should first find a "close" feasible point and then continue downhill.

4. The package requires only the evaluation of the objective function and its gradient. Subroutines for these evaluations are normally supplied by the user, although the gradient components may be obtained by differencing in some cases.

At least two such packages are now available [7,13]. We will refer to this package for linearly constrained problems as the LC package.

We now present the modified algorithm designed to achieve the practical objectives discussed above. The algorithm will be given first, followed by definitions of terms not previously defined. A more detailed discussion of the motivation for the modifications and remarks on the algorithm are given in Section 5.

Algorithm C

Input Data: matrix A; vectors b, $\underline{b}$, $\bar{b}$; functions ϕ_i, $i = 0,1,\ldots,q$, and their gradients; initial point x^0; tolerances ε, δ; parameters γ, $\bar{k}$. Also, any user specified tolerances and parameters required by the LC package.

1. Phase I: Generate the linearly constrained set $W_I(x^0)$, for input

to (PIM).

2. Call LC package and solve (PIM), starting with x^0. This gives x^1, and λ^1 as defined by (9). If $||x^1 - x^0|| \leq \varepsilon$, output (x^1,λ^1) as optimal vectors, and Stop.

3. Set $x^k \leftarrow x^1$, $\lambda^k \leftarrow \lambda^1$, $k \leftarrow 1$.

4. Phase II: Given x^k and λ^k, generate the linearly constrained set $W_{II}(x^k)$, for input to (PLM).

5. Call LC package and solve (PLM), starting with x^k. This gives x^{k+1} and $\lambda^{k+1} \geq 0$.

6. If $||x^{k+1} - x^k|| \leq \varepsilon$, output (x^{k+1},λ^{k+1}) as optimal vectors, and Stop.

7. If $||\phi^+(x^{k+1})|| \geq ||\phi^+(x^k)|| + \delta$, Phase II is not converging, Stop.

8. If $k = \bar{k}$, Stop.

9. Set $x^k \leftarrow x^{k+1}$, $\lambda^k \leftarrow \lambda^{k+1}$, $k \leftarrow k+1$, and return to step 4.

Definitions of terms used in Algorithm C:

The set of constraints linearized in Phase I depends on x^0, and is given by:

$$\bar{J}^0 \equiv \{i \mid |\phi_i(x^0)| \leq 10^6\delta\}$$

The linearly constrained set for Phase I is then defined as follows:

$$W_I(x^0) \equiv \left\{x \;\middle|\; \begin{matrix} x \in S \\ h_i(x^0;x) \leq 0,\ i \in \bar{J}^0 \end{matrix}\right\}$$

The problem solved in Phase I is given by:

$$\text{(PIM)} \qquad \min_{x \in W_I(x^0)} \bar{\Psi}(\bar{\mu};x)$$

where

$$\bar{\Psi}(\mu;x) \equiv \phi_0(x) + \frac{\mu}{2} \sum_{i=1}^{q} [\phi_i^+(x)]^2$$

and

$$\bar{\mu} = 2\gamma/10^7\delta$$

The solution to (PIM) gives x^1 and a set of multipliers $\bar{\lambda}_i \geq 0$, corresponding to those constraints $h_i(x^0;x) \leq 0$, $i \in \bar{J}^0$, which are active at x^1. We define the corresponding vector λ^1, with components given by

$$\lambda_i^1 \equiv \max\{\bar{\lambda}_i, \bar{\mu}\phi_i^+(x^1)\}\ ,\ i = 1,\ldots,q \tag{9}$$

The corresponding sets for Phase II are defined as follows:

$$J^j \equiv \{i \mid -10^6\delta \leq \phi_i(x^j)\}$$

$$J_c^k \equiv \bigcup_{j=1}^{k} J^j$$

$$W_{II}(x^k) \equiv \left\{x \;\middle|\; \begin{matrix} x \in S \\ h_i(x^k,x) \leq 0,\ i \in J_c^k \end{matrix}\right\}$$

The problem solved in the kth major iteration of Phase II is then given by:

$$\text{(PLM)} \qquad \min_{x \in W_{II}(x^k)} \Psi^k(x)$$

where

$$\Psi^k(x) \equiv \phi_0(x) + \sum_{i=1}^{q} \lambda_i^k\,[\phi_i(x) - h_i(x^k;x)]$$

The solution to (PLM) gives x^{k+1} and $\lambda^{k+1} \geq 0$. If the LC package exit from (PLM) occurs as a result of the specified iteration limit, the vector $\bar{\lambda}$ available may have some negative elements. In that case we let $\lambda_i^{k+1} \equiv \max\{0,\bar{\lambda}_i\}$.

5. Discussion

We conclude this paper with a more detailed discussion of the motivation for the modifications required in practical implementation of the algorithm, and also some additional remarks on the behavior of Algorithm C based on computational experience with a variety of test problems.

In order to achieve computational efficiency, it is important to use a proper value of μ in Phase I. A large value of μ will cause the Hessian matrix of $\bar{\Psi}$ to be ill-conditioned, resulting in a large increase in the computational effort (measured in either function calls or computer time) in Phase I. If too small a value is used, the point x^1 (and corresponding λ^1) obtained in Phase I will not be close enough to a local minimum (x^*,λ^*) to ensure convergence. Fortunately, there is a range within which any value of μ will be satisfactory. This is illustrated in Figure 1, which shows the (averaged and smoothed) behavior for some typical 15 variable problems using the GPM/NLC implementation. The three curves show the number of function calls required in Phase I, in Phase II, and the total in both. Each function call requires the evaluation of the function and its gradient corresponding to ϕ_0 and each of the active constraint functions ϕ_i. The Phase II curve includes all the iterations required in Phase II. As μ is increased, the x^1 and λ^1 given by Phase I become closer to x^* and λ^*, so that fewer Phase II iterations are required. The curve of total function calls is seen to have a relatively broad minimum, so that an order of magnitude change in μ from its best value would cause only a small increase in the computational effort required. For a different type and size of problem the shape of these curves would be similar, but the coordinate values would, in general, be different.

Roughly speaking, the best choice for μ is the smallest value which will produce an (x^1,λ^1) that gives convergence in Phase II. It is well known that one often gets convergence of Newton's method, as applied to solving a system of nonlinear equations, with a starting point which is reasonably good, but does not satisfy a sufficient condition for convergence. A similar situation may occur here.

We take advantage of this situation by choosing μ just large enough so that at the end of say $\hat{k}$ iterations in Phase II we will satisfy all nonlinear constraints within a specified tolerance δ. A satisfactory value of $\hat{k}$ would normally be $\hat{k} = 5$.

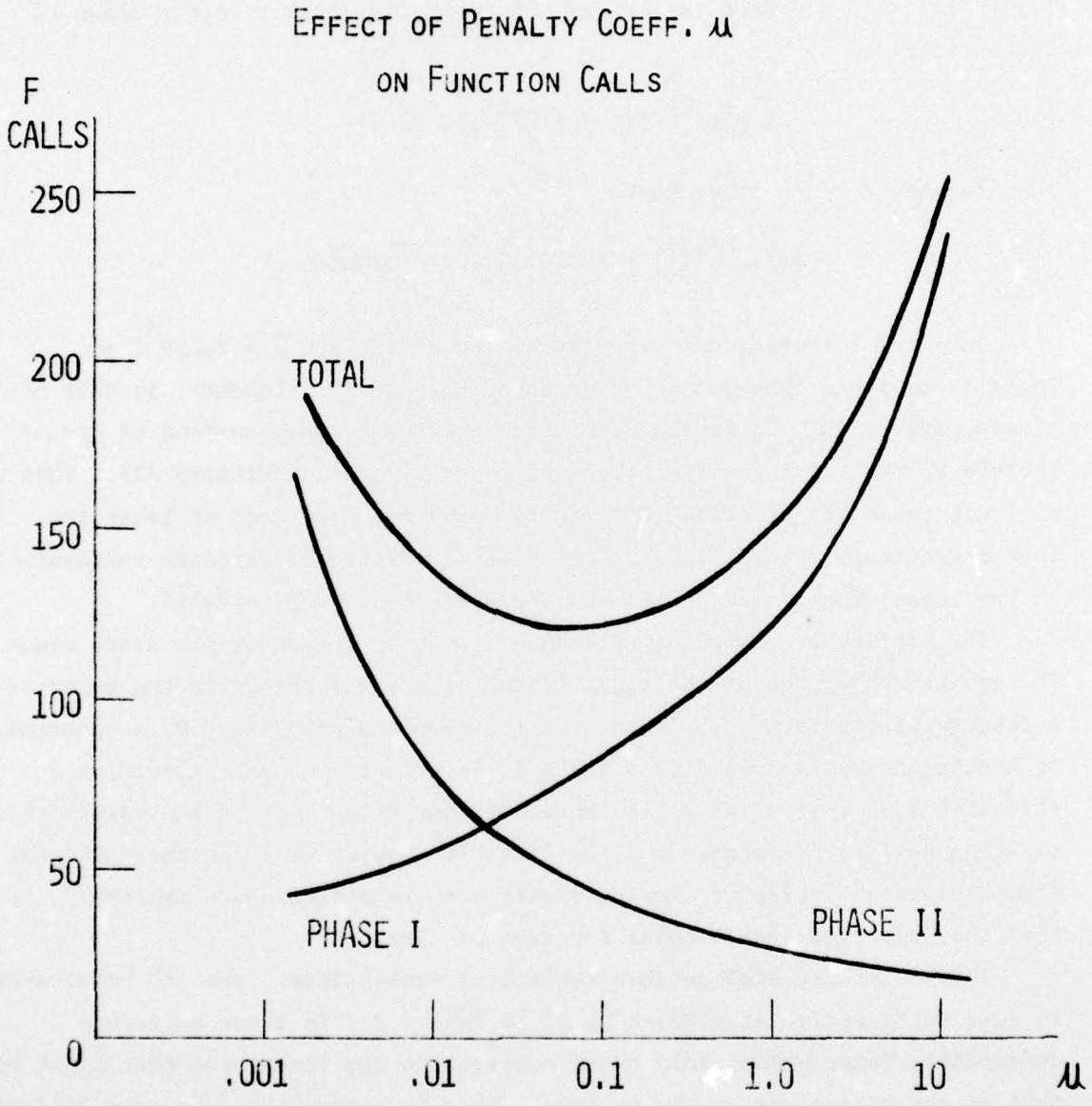
EFFECT OF PENALTY COEFF. μ
ON FUNCTION CALLS
F
CALLS
250
200
150
100
50
0
TOTAL
PHASE I
PHASE II
.001
.01
0.1
1.0
10
μ

Figure 1

From (7) we have

$$||\bar{\lambda}(\mu)|| = \mu||\phi^+(\bar{x}(\mu))||$$

where $\phi^+ \equiv (\phi_1^+,\ldots,\phi_q^+) \in R^q$. Since $\bar{\lambda}(\mu)$ is an approximation to λ^*, we have

$$||\phi^+(\bar{x}(\mu))|| \leq 2\gamma/\mu$$

where γ is an upper bound on $||\lambda^*||$. Now suppose we choose $\mu = \bar{\mu} = 2\gamma/10^7\delta$. Then at the end of Phase I the corresponding $x^1 = \bar{x}(\bar{\mu})$ will be such that $||\phi^+(x^1)|| \leq 10^7\delta$. From (4) we get the approximate relation for Phase II iterations:

$$||\phi^+(x^{k+1})|| \leq 2(\tfrac{1}{2})^{2^k}||\phi^+(x^1)|| \tag{10}$$

Thus for any $k \geq \hat{k} = 5$, we have

$$||\phi^+(x^{k+1})|| \leq 2\times 10^7(\tfrac{1}{2})^{2^k}\delta < 0.005\delta$$

Based on computational experience, the choice $\mu = \bar{\mu} = 2\gamma/10^7\delta$ in Phase I, achieves the desired convergence in Phase II. However, it will not always give an (x^1,λ^1) in the quadratic convergence neighborhood of (x^*,λ^*). Therefore, the first few iterations of Phase II may not satisfy (3). This will will not cause any practical difficulty, provided only that at least the last iteration of Phase II is taken in the quadratic convergence neighborhood. In that case, Theorem 2 applies and the error bound (8) is valid.

The nonlinear constraint tolerance δ must be chosen by the user, since it requires knowledge of the significance of a small change in the value of a constraint function ϕ_i. Instead of the requirement $\phi_i(x) \leq 0$, a constraint is assumed to be satisfied if $\phi_i(x) \leq \delta$. In effect, the user specifies a tolerance such that a change in the constraint value (or the equivalent, its right-hand-side) of magnitude δ, or less, is considered to be insignificant from a practical point of view. Constraints are scaled where necessary, so that the same tolerance δ holds for each of them.

This tolerance also defines the active constraints. The j^{th} constraint is said to be active at a point $\bar{x}$, if $|\phi_j(\bar{x})| \leq \delta$. In order to reduce unnecessary computation, only those constraints are linearized that might be part of the active set at the optimum. This is accomplished by only linearizing

those constraints which are either violated, or come close to being active at the end of Phase I, or any Phase II iteration. The justification for this is that if we knew in advance which nonlinear constraints were inactive at x^*, that is those for which $\phi_i(x^*) < -\delta$, we could eliminate them from consideration entirely without changing the problem (at least locally).

Another difficulty which can arise from linearizing a constraint $\phi_i(x)$ about $\bar{x}$, when $\phi_i(\bar{x})$ is negative and relatively large in magnitude, is shown by the following simple two variable problem. Let $\phi = -x_1^2 - x_2^2 + 1 \leq 0$, be the nonlinear constraint (outside of circle is feasible). Also assume the bound $x_1 \leq 1.5$. Any point (x_1,x_2) outside the circle of unit radius and with $x_1 \leq 1.5$, is of course feasible. However, it is easy to see that linearizing ϕ about $(\bar{x}_1,\bar{x}_2) = (3,0)$ gives $x_1 \geq 5/3$, so that there is no feasible solution to the linearized problem.

On the other hand, it is useful to linearize some constraints even in Phase I, if we wish to recognize a local minimum (or a point close to a local minimum) when such a point is used as the initial point x^0 in Phase I. This difficulty is resolved by linearizing all those constraints which are close to x^0, more specifically those ϕ_i for which $i \in \bar{J}^0$. Even if a constraint, say ϕ_j, has been linearized in Phase I and in addition is active at x^1 (with $\bar{\lambda}_j > 0$), it is likely that we will have $\phi_j(x^1) > 0$. In view of this possibility, the equation (9) gives the best estimate for λ^* at the end of Phase I.

We now conclude with some remarks which may be helpful in understanding Algorithm C.

1. If x^0 is a local minimum, then all nonlinear constraints active at this minimum (i.e., those with $\phi_i(x^0)=0$) will be linearized and included in $W_I(x^0)$. Therefore, x^0 will be an optimal solution to (PIM), so that it will give $x^1=x^0$. Thus, if the initial point is a local minimum, it will be recognized as such at the start of Phase I.

2. The initial point x^0 need not be feasible. If the original linear constraint set S has no feasible solution, this will be recognized by the LC package at the start of Phase I (i.e., a no feasible solution exit from LC).

3. If S is not empty, the sequence of points $\{x^k\}$ generated in Phase I and II will all be in S. However, even if $x^0 \in V$, we will usually not

have $x^k \in V$, $k = 1,2,\ldots,\bar{k}$. Thus, in general, $||\phi^+(x^k)|| > 0$, although this constraint violation norm will go to zero quadratically as given by (10).

4. The LC package will legitimately stop for one of three reasons: (a) no feasible solution exists to the linear constraints; (b) an optimal solution is obtained within the LC tolerances; (c) the LC iteration limit is reached.

If (a) occurs it indicates no feasible solution exists to (PN), or other serious difficulty, and the algorithm terminates. If (b) or (c) occur, the vectors x^{k+1} and λ^{k+1} are available and the algorithm continues.

5. Successful termination of the algorithm occurs only in Step 2 or Step 6.

6. If the algorithm terminates in Step 7 or 8, the vectors x^{k+1} and λ^{k+1} are available, but in general are not optimal.

7. A constraint function ϕ_i and its gradient is only called in (PIM) when $\phi_i^+ > 0$. Similarly in (PLM), only when $\lambda_i^k > 0$. This can significantly reduce the amount of computation required.

8. The constants 10^6 used in the definition of $\bar{J}^0$ and J^j, and 10^7 used in computing $\bar{\mu}$ were determined as best so as to minimize the computational effort in a variety of test problems. A change in these values by a factor of 10 seems to have little effect.

References

1. A. Buckley, "An alternate implementation of Goldfarb's minimization algorithm", Math. Prog. 8 (1975), pp. 207-231.

2. A. V. Fiacco and G. P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, Wiley (1968). Chap. 4.

3. P. E. Gill and W. Murray, "Newton-type methods for unconstrained and linearly constrained optimization", Math. Prog. 7 (1974), pp. 311-350.

4. A. S. Manne, "ETA-MACRO: A model of energy-economy interactions"; Presented at ORSA/TIMS Meeting, San Francisco, May 9-11, 1977.

5. R. Meyer, "The validity of a family of optimization methods", SICON 8 (1970), pp. 41-54.

6. B. A. Murtagh and M. A. Saunders, "Nonlinear programming for large, sparse system", Tech. Rept. SOL 76-15, Systems Optimization Lab., Stanford, Aug. 1976.

7. B. A. Murtagh and M. A. Saunders, "MINOS, A large scale nonlinear programming system", Tech. Rept. SOL 77-9, Systems Optimization Lab., Stanford, Feb. 1977.

8. M. J. D. Powell, "Algorithms for nonlinear constraints that use Lagrangian functions", to appear in Proceedings of Ninth Symposium on Mathematical Programming, Budapest, Aug. 1976.

9. S. M. Robinson, "A quadratically convergent algorithm for general nonlinear programming problems", Math. Prog. 3 (1972), pp. 145-156.

10. S. M. Robinson and R. R. Meyer, "Lower semicontinuity of multivalued linearization mappings", SIAM J. Control 11 (1973), pp. 525-533.

11. J. B. Rosen, "Iterative solution of nonlinear optimal control problems", SICON 4 (1966); pp. 223-244.

12. J. B. Rosen and J. Kreuser, "A gradient projection algorithm for nonlinear constraints", Numerical Methods for Nonlinear Optimization (F. A. Lootsma, Ed.) Academic Press (1972), pp. 297-300.

13. J. B. Rosen and S. Wagner, "The GPM nonlinear programming subroutine package; description and user instructions", Tech. Rept. 75-9, Computer Science Dept., Univ. of Minnesota, May 1975.

14. J. B. Rosen, "A two-phase algorithm for nonlinear constraint problems"; Presented at Symposium on Matrix Methods in Optimization, Argonne National Lab, June 14-15, 1976.

15. J. B. Rosen and N. Zilverberg, "Computational results with a two-phase algorithm for nonlinear constraint programming problems". Technical Report in preparation.

16. Margaret Wright, "Numerical methods for nonlinearly constrained optimization", Technical Report SLAC-193, Stanford Univ., March 1976.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: (14) TR - 77-22
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): (6) Two-Phase Algorithm for Nonlinear Constraint Problems.
5. TYPE OF REPORT & PERIOD COVERED: (9) Technical Report.
6. PERFORMING ORG. REPORT NUMBER: 77-22

7. AUTHOR(s): (10) J.B. Rosen
8. CONTRACT OR GRANT NUMBER(s): (15) N00014-75-C-0267

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Department of Operations Research (SOL)
Stanford University
Stanford, CA 94305

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR-047-064

11. CONTROLLING OFFICE NAME AND ADDRESS:
Operations Research Program
Office of Naval Research
Arlington, VA 22217

12. REPORT DATE: Aug[illegible]77
13. NUMBER OF PAGES: 22 (12) 27p.

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office):

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

NONLINEAR PROGRAMMING
COMPUTATIONAL METHOD
QUADRATIC CONVERGENCE
NONLINEAR CONSTRAINTS
LINEARIZATION
ENERGY MODEL

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See attached

over

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

402 766

77-22 "Two-Phase Algorithm for Nonlinear Constraint Problems"

An algorithm is described which solves the general nonlinear programming problem with nonlinear constraints. The computational implementation is based on the iterative use of any available package which solves the linearly constrained problem with a nonlinear objective function. Large, sparse problems with nonlinear constraints can be solved by this algorithm, provided a suitable linear constraint package is used.

The algorithm consists of two phases. The first (Phase I) uses an external squared penalty function to find a point $x^{(1)}$, close to a local minimum. Starting with $x^{(1)}$, the algorithm then solves a sequence of linearly constrained problems (Phase II). Selected nonlinear constraints are linearized for each such Phase II iteration. With suitable assumptions, convergence from any initial point, with quadratic convergence in Phase II, is shown.

The practical implementation of this algorithm is described, and its potential application to a model for the assessment of energy alternatives is discussed briefly.